UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/872,091	06/01/2001	Yasuteru Araya	14669	2464

23389 7590 02/01/2007

SCULLY SCOTT MURPHY & PRESSER, PC
400 GARDEN CITY PLAZA
SUITE 300
GARDEN CITY, NY 11530

EXAMINER	
THANGAVELU, KANDASAMY	
ART UNIT	**PAPER NUMBER**
2123	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/01/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	09/872,091	ARAYA ET AL.	
	Examiner	Art Unit	
	Kandasamy Thangavelu	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 December 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *11-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *11-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *01 June 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This communication is in response to the Applicants' Amendment dated December 7, 2006. Claims 11 and 16 were amended. Claims 11-26 of the application are pending. This office action is made non-final.

Claim Rejections - 35 USC § 101

2. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3. Claims 11-26 are rejected under 35 U.S.C. 101 because the claimed inventions are directed to non-statutory subject matter.

This is because the claims do not produce any useful, credible and tangible results. Claim 11 does several computer operations terminating in performing the performance evaluation by simulating the modified source code elements and counting the data traffic on the bus. Claim 20 does several computer operations terminating in performing the evaluation of the performance of the bus in response to the bus traffic being produced. To produce useful, tangible and credible results, they should use the results of the performance evaluation to modify the design of the LSI and should indicate what is produced by the design process, for example a file specifying the isolation of the LSI design into hardware units and software units.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

5. The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

6. Claims 11-19 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Tammemae et al.** ("AKKA: A tool for cosynthesis and prototyping", The Institution of Electrical Engineers, UK, 1996) in view of **Raimi et al.** (U.S. Patent 5,604,895).

6.1 **Tammemae et al.** teaches AKKA: A tool for cosynthesis and prototyping. Specifically, as per claim 11, **Tammemae et al.** teaches a method in a LSI design and development process (Page 1, Para 1, L1-2; Page 1, Para 2, L2-5), for evaluating an architecture design for an

algorithm design by performing a performance evaluation of at least one bus at a high-level stage of the design and development process (Page 1, Para 2, L2-4; Page 1, Para 4, L2; Page 2, Para 4, L3); the method comprising:

structuring source code describing the algorithm design in a general purpose high-level programming language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3), by isolating elements of the source code representing hardware units and software units (Page 1, Para 2, L3; Page 1, Para 3, L7-8; Page 3, Fig. 1);

creating an evaluation function for counting data traffic that occurs on the at least one bus (Page 1, Para 4, L2; Page 1, Para 2, L3-4; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 3, Fig. 1: compile for data transfer profiling; execute to get data transfer profiling information; Page 2, Para 2, L4), the bus being a part of the source code realizing the data traffic between the elements representing hardware units and software units (Page 1, Para 4, L2; Page 1, Para 2, L3-4; Page 1, Para 5, L5-6: the bus in the co-simulation is part of the source code realizing data traffic between the elements representing hardware units and software units; Page 2, Para 2, L4; Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 3, Fig. 1: compile for data transfer profiling; execute to get data transfer profiling information);

modifying at least one element of the source code elements based on a result of an implementation of the evaluation function (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 2, L4: additional profiling functions inserted into an executable code);

performing the performance evaluation by simulating the modified source code elements and counting the data traffic on the bus (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 3, Fig. 1: HW/SW compiling and co-verification; Page 4, Para 1, L2-3).

Tammemae et al. teaches that during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable (Page 2, Para 5, L1-3). **Tammemae et al.** does not expressly teach sequentially reading in the source code line by line while effecting syntax analysis; determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated. **Raimi et al.** teaches sequentially reading in the source code line by line while effecting syntax analysis (Abstract, L5-10: the CPU parses the high level language description, to allow for generation of new code; CL2, L20-25: the method begins by parsing the original high level language description having at least one executable statement to obtain information; a high level language description, which includes new code generated in

response to the information and the code of the original high level language description is generated); determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated (CL1, L26-32: profiling is the practice of surrounding the existing computer code of a high level language program with monitor code; profiling is synonymous with instrumenting; CL1, L64-66: profiling is generation of additional code to check for the occurrence of various events; CL2, L37-54: a bus is coupled to the memory unit; a central processing unit (CPU) coupled to the bus to allow communication between the CPU and memory; the bus referred to here the bus of the development system which is a real bus; the CPU reads the original high level language description from the memory unit and creates new high level language description; the new high level language description includes a plurality of new assignment statements). It would have been obvious to one of ordinary skill in the art at the time of Applicants' invention to modify the method of **Tammemae et al.** with the method of **Raimi et al.** that included sequentially reading in the source code line by line while effecting syntax analysis; determining whether the source code was to be modified based on whether a line of source code represented writing data to variables that were defined in advance and were loaded onto the bus to be evaluated because that would allow generation of additional code to check for the occurrence of bus access events (CL1, L65-66).

Per claim 12: **Tammemae et al.** teaches restructuring the source code based on the evaluated data traffic (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2); and

performing the performance evaluation again by simulating the restructured source code again (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1).

Per claim 13: **Tammemae et al.** teaches that a bus traffic is calculated from the evaluated data traffic with respect to the processing rate of the bus (Page 2, Para 5, L1-3).

Per claim 14: **Tammemae et al.** teaches feeding back a result of the performance evaluation of the bus to the step of structuring the source code to improve the architecture design at a high-level design stage by isolating in the source code new elements representing hardware units and new elements representing software units (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2; Page 2, Para 5, L1-3).

Per claim 15: **Tammemae et al..** teaches that in response to the bus traffic, isolation of the source code into elements representing hardware units and elements representing software units is optimized (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2; Page 2, Para 5, L1-3).

6.2 As per claim 16, **Tammemae et al.** and **Raimi et al.** teach the method of claim 11. **Tammemae et al.** teaches profiling the source code based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated Page 2, Para 2, L4: additional profiling functions inserted into an executable code; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 5, L1-3: during data transfer profiling for

each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file);

structuring the source code into elements representing at least one of the hardware units and the software units for use in the architecture design by compiling the source code (Page 1, Para 2, L3; Page 1, Para 3, L7-8; Page 3, Fig. 1);

calculating the data transfer rate on the bus by executing the compiled source code elements in a simulation program (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made; Page 3, Fig. 1; Page 4, Para 1, L2-3);

calculating bus traffic with regard to a given processing rate of the bus (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made); and

performing evaluation of the performance of the bus in response to the bus traffic (Page 2, Para 5, L1-3; Page 3, Fig. 1: HW/SW compiling and co-verification; Page 4, Para 1, L2-3).

Tammemae et al. does not expressly teach upon determining that the source code is to be modified, modifying the source code by embedding the evaluation function one of immediately before or immediately after the line of source code in which the variable is written; and repeating the forgoing steps until the source code is completely read in up to a last line of source code. **Raimi et al.** teaches upon determining that the source code is to be modified, modifying the source code by embedding the evaluation function one of immediately before or immediately after the line of source code in which the variable is written (CL1, L26-32: profiling is the practice of surrounding the existing computer code of a high level language program with

monitor code; profiling is synonymous with instrumenting; CL1, L64-66: profiling is generation of additional code to check for the occurrence of various events; CL2, L20-25: the method begins by parsing the original high level language description having at least one executable statement to obtain information; a high level language description, which includes new code generated in response to the information and the code of the original high level language description is generated; CL2, L37-54: a bus is coupled to the memory unit; a central processing unit (CPU) coupled to the bus to allow communication between the CPU and memory; the bus referred to here the bus of the development system which is a real bus; the CPU reads the original high level language description from the memory unit and creates new high level language description; the new high level language description includes a plurality of new assignment statements); and repeating the forgoing steps until the source code is completely read in up to a last line of source code (Abstract, L5-6; CL2, L20-22).

Per claim 17: **Tammemae et al.** teaches that the variables loaded onto the bus consist of n bits while the bus consists of m bit lines, where n and m are both integers, and n is a multiple of m, and the bus traffic for the processing rate is produced such that the number of times in effecting data transfer on the bus is multiplied by n/m and is then divided by the processing rate (Page 2, Para 5, L1-3). **Tammemae et al.** teaches that each variable access is counted using a counter and a log of the access of the variables is made. It is inherent that from the counting of the variable access to a bus, and the log of the variable accesses, it is possible to calculate the number of times the bus was used if the bus width was smaller than the variable length requiring multiple accesses to the bus for each variable loaded onto the bus.

Per claim 18: **Tammemae et al.** teaches that the general purpose high-level language is one of C language and C++ language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3).

Per claim 19: **Tammemae et al.** teaches that the evaluation function increments a counting value if a pre-defined variable is loaded onto the bus (Page 2, Para 5, L1).

6.3 As per claim 20, **Tammemae et al.** teaches a method in a LSI design and development process (Page 1, Para 1, L1-2; Page 1, Para 2, L2-5), for evaluating and facilitating an architectural design for an algorithm design by performing a performance evaluation of at least one bus at the architecture design being a high-level stage of the design and development process (Page 1, Para 2, L2-4; Page 1, Para 4, L2; Page 2, Para 4, L3); the method comprises the steps of:

structuring source code describing the algorithm design in a general purpose high-level programming language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3), by isolating the source code in hardware and software elements interconnected by the bus having a given bus configuration (Page 1, Para 2, L3-4; Page 1, Para 3, L7-8; Page 1, Para 4, L2; Page 1, Para 5, L5-6; Page 3, Fig. 1);

creating an evaluation function for evaluating data transfer that occurs on the bus (Page 1, Para 4, L2; Page 1, Para 2, L3-4; Page 2, Para 4, L3; Page 3, Fig. 1; Page 2, Para 2, L4), wherein the evaluation function counts and represents a number of times the source code effecting the data transfer on the bus (Page 2, Para 5, L1);

simulating the modified source code elements at the architecture design level and evaluating the data transfer on the bus (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1, L2-3);

compiling the structured source code elements (Page 1, Para 4, L1-2);

executing the compiled source code elements on a simulation platform (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1, L2-3);

calculating the bus traffic based on the number of times in effecting of data transfers according to the evaluation function and a given processing rate that is already known (Page 2, Para 5, L1-3);

performing the evaluation of the performance of the bus in response to the bus traffic being produced (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1, L2-3); and

wherein the method is carried out again in response to the performance evaluation if necessary by starting with changing the bus configuration and restructuring the source code (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1).

Tammemae et al. does not expressly teach sequentially reading in the source code; determining whether a line of source code represents writing data onto the bus to be evaluated; upon the determination, modifying the source code by embedding the evaluation function just before or after the line of source code in which the variable is written; and repeating the forgoing steps until the source code is completely read in and modified up to a last line of source code. **Raimi et al.** teaches sequentially reading in the source code (Abstract, L5-10; CL2, L20-25); determining whether a line of source code represents writing data onto the bus to be evaluated

(CL1, L26-32; CL1, L64-66; CL2, L37-54); upon the determination, modifying the source code by embedding the evaluation function just before or after the line of source code in which the variable is written (CL1, L26-32; CL1, L64-66; CL2, L23-29; CL2, L37-54); and repeating the forgoing steps until the source code is completely read in and modified up to a last line of source code (Abstract, L5-6; CL2, L20-22).

Per claim 21: **Tammemae et al.** teaches feeding back a result of the performance evaluation of the bus to the step of structuring the source code to improve the architecture design at a high-level design stage by re-isolating the source code in hardware and software elements (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2; Page 2, Para 5, L1-3).

Per claim 23: **Tammemae et al.** does not expressly teach after creating the evaluation function, sequentially reading in the source code line by line while effecting syntax analysis. **Raimi et al.** teaches after creating the evaluation function, sequentially reading in the source code line by line while effecting syntax analysis (Abstract, L5-10; CL2, L20-252).

Per claim 24: **Tammemae et al.** teaches that the variables loaded onto the bus consist of n bits while the bus consists of m bit lines, where n and m are both integers, and n >= m, and the bus traffic for the processing rate is produced such that the number of times in effecting data transfer on the bus is multiplied by n/m and is then divided by the processing rate (Page 2, Para 5, L1-3). **Tammemae et al.** teaches that each variable access is counted using a counter and a log of the access of the variables is made. It is inherent that from the counting of the variable

access to a bus, and the log of the variable accesses, it is possible to calculate the number of times the bus was used if the bus width was smaller than the variable length requiring multiple accesses to the bus for each variable loaded onto the bus.

Per claim 25: **Tammemae et al.** teaches that the general purpose high-level language is one of C language and C++ language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3).

Per claim 26: **Tammemae et al.** teaches that the evaluation function is to increment a counting value if a pre-defined variable is loaded onto the bus (Page 2, Para 5, L1).

7. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Tammemae et al.** ("AKKA: A tool for cosynthesis and prototyping", The Institution of Electrical Engineers, UK, 1996) in view of **Raimi et al.** (U.S. Patent 5,604,895), and further in view o **Chang et al.** (U.S. Patent 6,269,467).

7.1 As per claim 22, **Tammemae et al.** and **Raimi et al.** teach the method of claim 20. **Tammemae et al.** teaches that the bus configuration comprises the number of bit lines of the bus (Page 2, Para 5, L1-3); and isolation of the source code in hardware and software elements is optimized (Page 1, Para 2, L3; Page 1, Para 3, L7-8; Page 3, Fig. 1).

Tammemae et al. and **Raimi et al.** do not expressly teach in response to the bus traffic changing the number of bit lines of the bus. **Chang et al.** teaches in response to the bus traffic

changing the number of bit lines of the bus (CL27, L18-27; CL29, L29-35; CL33, L53-59; CL39, L1-7).

Response to Arguments

8. Applicants' arguments with respect to 35 USC 103 (a) rejections are not persuasive. In addition, Applicants' amendments to claims 11 and 16 have necessitated using new 103 (a) rejections for those claims. Additional claim rejections under 35 USC 101 are included in this Office Action.

8.1 As per the applicants' argument that "The Examiner contends that Raimi et al. teaches the steps of reading in the source code line by line while effecting syntax analysis, and also determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated, and that it would have been obvious to one of ordinary skill in the art to modify the combination of Tammemae and Chang with Raimi", the Examiner has used Tammemae with Raimi in response to the Applicants' amendment to the claims.

Tammemae et al. teaches determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a

function which writes the logged data to the file; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 2, L4: additional profiling functions inserted into an executable code).

Tammemae et al. teaches modifying at least one element of the source code elements based on a result of an implementation of the evaluation function; the virtual bus between the hardware and software elements in the system being modeled is in the source code elements and is being monitored as part of data profiling (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 2, L4: additional profiling functions inserted into an executable code).

Raimi et al. teaches sequentially reading in the source code line by line while effecting syntax analysis (Abstract, L5-10: the CPU parses the high level language description, to allow for generation of new code; CL2, L20-25: the method begins by parsing the original high level language description having at least one executable statement to obtain information; a high level language description, which includes new code generated in response to the information and the code of the original high level language description is generated); determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated (CL1, L26-32: profiling is the practice of surrounding the existing computer code of a high level language program with monitor code; profiling is synonymous with instrumenting; CL1, L64-66: profiling is generation of additional code to check for the occurrence of various events; CL2, L37-54: a bus is coupled to the memory unit; a central processing unit (CPU) coupled to the bus to allow

communication between the CPU and memory; the bus referred to here the bus of the development system which is a real bus; the CPU reads the original high level language description from the memory unit and creates new high level language description; the new high level language description includes a plurality of new assignment statements). It would have been obvious to one of ordinary skill in the art at the time of Applicants' invention to modify the method of **Tammemae et al.** with the method of **Raimi et al.** that included sequentially reading in the source code line by line while effecting syntax analysis; determining whether the source code was to be modified based on whether a line of source code represented writing data to variables that were defined in advance and were loaded onto the bus to be evaluated because that would allow generation of additional code to check for the occurrence of bus access events (CL1, L65-66).

8.2 As per the applicants' argument that "Raimi simply teaches the bus definition realizing the input/output operation of data with regard to the memory (20) because Raimi can perform evaluation or estimation of data sent via the bus for the purpose of the evaluation of the electrical circuit, but not for the purpose of optimization of the bus. Hence no description regarding the bus optimization is found in' the specification. Therefore, the physical bus disclosed by Raimi is distinct from the virtual bus of the present invention", the Examiner agrees that the bus discussed in Rahimi is the physical bus of the development system that is used by the CPU to get the original source code from memory and put the modified source code into the memory. However, the bus being modeled between the hardware and software elements of the system being modeled is taught by **Tammemae et al.**

Tammemae et al. teaches determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 2, L4: additional profiling functions inserted into an executable code).

Tammemae et al. teaches modifying at least one element of the source code elements based on a result of an implementation of the evaluation function; the virtual bus between the hardware and software elements in the system being modeled is in the source code elements and is being monitored as part of data profiling (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 2, L4: additional profiling functions inserted into an executable code).

8.3 As per the applicants' argument that "because Raimi does not teach or suggest evaluating the performance of a bus, he does not teach determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated, as recited in claim 11", the Examiner respectfully disagrees. Applicants' attention is directed to paragraph 8.1 above.

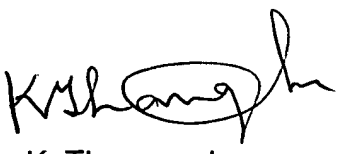
Conclusion

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dr. Kandasamy Thangavelu whose telephone number is 571-272-3717. The examiner can normally be reached on Monday through Friday from 8:00 AM to 5:30 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Paul Rodriguez, can be reached on 571-272-3753. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



K. Thangavelu
Art Unit 2123
January 29, 2007